CLAIMS

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is as follows:

1. A method comprising:

employing a computer for:

obtaining a collection of code;

providing a program graph representing said collection of code;

identifying any authorization resources of said collection of code;

locating any bounded path within said program graph; and

associating said any authorization resource with said any bounded path.

2. A method as recited in claim 1, wherein said collection of code includes codes obtained from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

3. A method as recited in claim 1, wherein the step of providing includes constructing said program graph through static analysis techniques.

4. A method as recited in claim 3, wherein the step of constructing includes employing object code.

5. A method as recited in claim 1, wherein the step of identifying includes finding at least one authorization point in said program graph.

6. A method as recited in claim 5, wherein the step of finding includes finding a Java authorization point.

7. A method as recited in claim 6, wherein the step of find a Java authorization point includes finding a AccessController.checkPermission node in said program graph, and finding a java.security.Permission object passed as an argument to the AccessController.checkPermission method.

8. A method as recited in claim 5, wherein said authorization point is an instruction invocation.

9. A method as recited in claim 8, wherein said instruction invocation is used in a particular language for said collection of code.

10. A method as recited in claim 9, wherein said particular language is C#.

11. A method as recited in claim 1, wherein the step of identifying includes employing data flow analysis.

12. A method as recited in claim 11, wherein the step of employing includes generating a data flow from said program graph.

13. A method as recited in claim 1, wherein the step of identifying any bounded path includes locating a set of start nodes in said program graph, and locating a stop node in said program graph; and said bounded path includes all nodes within the graph bound by said start nodes and said stop node.

14. A method as recited in claim 1, wherein the step of associating includes associating and aggregating said any authorization resource with said collection of code.

15. An apparatus comprising:

computing means having:

means for obtaining a collection of code;

means for providing a program graph representing said collection of code;

means for identifying any authorization resources of said collection of code;

means for locating any bounded path within said program graph; and

means for associating said any authorization resource with said any bounded path.

16. An apparatus as recited in claim 15, wherein said collection of code includes codes obtained from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

17. An apparatus as recited in claim 15, wherein the means for providing includes means for constructing said program graph through static analysis techniques.

18. An apparatus as recited in claim 17, wherein the means for constructing includes employing object code.

19 . An apparatus as recited in claim 15, wherein the means for identifying includes means for finding at least one authorization point in said program graph.

20. An apparatus as recited in claim 19, wherein the means for finding includes finding a Java authorization point.

21. An apparatus as recited in claim 20, wherein the means for find a Java authorization point includes finding a AccessController.checkPermission node in said program graph, and finding a

java.security.Permission object passed as an argument to the AccessController.checkPermission method.

22. An apparatus as recited in claim 19, wherein said authorization point is an instruction invocation.

23. An apparatus as recited in claim 22, wherein said instruction invocation is used in a particular language for said collection of code.

24. An apparatus as recited in claim 23, wherein said particular language is C#.

25. An apparatus as recited in claim 15, wherein the means for identifying includes employing data flow analysis.

26. An apparatus as recited in claim 25, wherein the means for employing includes generating a data flow from said program graph.

27. An apparatus as recited in claim 15, wherein the means for identifying any bounded path includes locating a set of start nodes in said program graph, and locating a stop node in said program graph; and said bounded path includes all nodes within the graph bound by said start nodes and said stop node.

28. An apparatus as recited in claim 15, wherein the means for associating includes associating and aggregating said any authorization resource with said collection of code.

29. A method comprising:

employing a computer including the steps of:

obtaining a collection of code;

providing a program graph representing said collection of code; and

identifying a complete set of authorization resources of said collection of code.

30. A method as recited in claim 29, if no resource is identified in said step of identifying, further comprising providing an indication that authorization testing is not necessary.

31. A method as recited in claim 29, further comprising:

identifying any bounded path within said program graph; and

associating any authorization resource identified in the step of identifying with said any bounded path.

32. An apparatus comprising:

a computer having access to a collection of code, the computer including:

an authorization resource identifier to identify any authorization resources within the collection of code;

a bounded path locator to locate any bounded path within a program graph of said collection of code; and

an associator to associate said any authorization resource with said any bounded path.

33. An apparatus as recited in claim 32, wherein said collection of code includes codes obtained from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

34. An apparatus as recited in claim 32, further comprising a program graph constructor to construct said program graph through static analysis techniques.

35. An apparatus as recited in claim 32, wherein the program graph constructor uses object code of said collection of code.

36. An apparatus as recited in claim 32, wherein the authorization resource identifier finds at least one authorization point in said program graph.

37. An apparatus as recited in claim 36, wherein at least one authorization point of said at least one authorization point is a Java authorization point.

38. An apparatus as recited in claim 37, wherein the Java authorization point includes finding a AccessController.checkPermission node in said program graph, and finding a java.security.Permission object passed as an argument to the AccessController.checkPermission method.

39. An apparatus as recited in claim 36, wherein said authorization point is an instruction invocation.

40. An apparatus as recited in claim 39, wherein said instruction invocation is used in a particular language for said collection of code.

41. An apparatus as recited in claim 40, wherein said particular language is C#.

42. An apparatus as recited in claim 32, wherein the authorization resource identifier employs data flow analysis.

43. An apparatus as recited in claim 42, wherein the data flow analysis is generated from said program graph.

44. An apparatus comprising:

a computing module having access to a collection of code, the computer module including:

an authorization resource identifier to completely identify any authorization resources within a collection of code; and

a bounded path locator to locate any bounded path within a program graph of said collection of code.

45. An apparatus as recited in claim 44, wherein the authorization resource identifier provides an indication that authorization testing is not necessary if no resource is identified by said authorization resource identifier.

46. An apparatus as recited in claim 44, further comprising an associator to associate said any authorization resource with said any bounded path.

47. An apparatus comprising:

a computer including:

means for obtaining a collection of code;

means for providing a program graph representing said collection of code; and

means for identifying a complete set of authorization resources of said collection of code.

48. An apparatus as recited in claim 47, further comprising means for indicating if authorization testing is necessary.

49. An apparatus as recited in claim 47, further comprising:

means for identifying any bounded path within said program graph; and

means for associating any authorization resource identified in the step of identifying with said any bounded path.

50. A method comprising:

employing a computer for:

constructing a program graph from a collection of code using static analysis techniques;

performing a data flow analysis using static analysis techniques;

searching said program graph for any useful resource for executing said collection of code;

identifying any bounded path within said program graph over which said useful resource is beneficial; and

associating said useful resource with said collection of code.

51. The method as recited in claim 50, wherein the step of constructing includes employing source code of said collection of code.

52. A method as recited in claim 50, wherein the step of constructing includes building an invocation graph of said collection of code to form said program graph.

53. A method as recited in claim 50, wherein the step of constructing a program graph includes constructing a call graph of said collection of code to form said program graph.

54. A method as recited in claim 50, wherein said step of constructing includes using context-sensitivity.

55. A method as recited in claim 54, wherein said step of using context sensitivity includes using type information for any method receiver and/or any parameter.

56. A method as recited in claim 55, wherein the step of using type information includes using class and memory allocation site information.

57. A method as recited in claim 56, wherein the step of using type information includes using per instance information.

58. A method as recited in claim 57, wherein the step of using instance information includes associating instance information with a node or edge in said program graph.

59. A method as recited in claim 50, wherein said collection of code includes codes constructed from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

60. A method as recited in claim 50, wherein the step of searching includes locating a node or edge in said program graph that represents a location where said useful resource would be beneficial.

61. A method as recited in claim 60, wherein said useful resource is a resource identifier.

62. A method as recited in claim 60, wherein said location is an authorization test.

63. A method as recited in claim 50, wherein said program graph represents an object oriented program.

64. A method as recited in claim 50, wherein said program graph and said data flow analysis are constructed from Java object code.

65. A method as recited in claim 61, wherein said resource identifier includes at least one Java java.security.Permission.

66. A method as recited in claim 62, wherein said authorization test is a call to any java.security.AccessController.checkPermission method.

67. A method as recited in claim 62, wherein said location represents a call to any authorization testing method in any instance of java.lang.SecurityManager and/or one of its subclasses.

68. A method as recited in claim 55 , wherein said node has a parameter which said type information is a Java java.security.Permission.

69. A method as recited in claim 68, wherein the step of identifying includes locating the constructor for said Java java.security.Permission allocation site and using said data flow analysis in identifying any value passed by any parameter to said constructor, wherein the combination of the Java java.security.Permission and a value for any parameter is the used Permission.

70. A method as recited in claim 50, wherein the step of identifying any bounded path includes locating a set of start nodes in said program graph, and locating a stop node in said program graph; and said bounded path includes all nodes within the graph bound by said start nodes and said stop node.

71. A method as recited in claim 50, wherein the step of associating includes mapping a subset of said collection of code to said useful resource.

72. A method as recited in claim 71, wherein the subset of said collection of code includes nodes in said bounded path.

73. A method as recited in claim 72, further comprising employing a privileged Java code wherein said stop node represents the method java.security.AccessController.checkPermission(); and

employing said start nodes are any of root nodes in said program graph or a node representing the method java.security.AccessController.doPrivileged().

74. A method as recited in claim 73, further comprising connecting a used Permission with said privileged Java code.

75. A method as recited in claim 74, further comprising connecting a used Permission with any node in said program graph prior to said java.security.AccessController.doPrivileged() node.

76. A method as recited in claim 74, wherein said step of associating includes connecting a used Permission for each java.security.AccessController.checkPermission() in said program graph.

77. A method as recited in claim 76, wherein said step of associating includes connecting said used Permission from each node in said program graph to each method associated with said program graph.

78. A method as recited in claim 77, wherein said step of associating includes connecting said used Permission from each method in said program graph to each class in said program graph.

79. A method as recited in claim 78, wherein said step of associating includes connecting said used Permission from each class in said program graph to collection of classes.

80. A method as recited in claim 79, wherein said step of associating includes connecting said used Permission is associated from each class in said program graph to collection of classes.

81. A method as recited in claim 50, further comprising employing said useful resource in executing said collection of code.

82. A method comprising:

statically detecting useful resources for a collection of code written in a computer programming language, by including the steps of:

calculating if any code of said collection of code is part of a program graph;

identifying any useful resource for said collection of code;

determining any bounded path of nodes within said program graph which constrain said useful resources of said collection of code; and

associating said any useful resource with said collection of code within said bounded path of nodes in said program graph.

83. A method as recited in claim 82, wherein the step of calculating includes using an invocation graph having a set of root nodes for said program graph.

84. A method as recited in claim 82, wherein the step of calculating includes using a call graph having a set of root nodes for said program graph.

85. A method as recited in claim 83, further comprising determining whether a basic block, method, class is represented as a node in said invocation graph.

86. A method as recited in claim 84, further comprising determining whether a basic block, method, class is represented as a node in said call graph.

87. A method as recited in claim 83, further comprising identifying a node in said invocation graph that identifies said useful resources, wherein said useful resources is one or more resources.

88. A method as recited in claim 84, further comprising a step identifying a node in said invocation graph that identifies said useful resources, wherein said useful resources is one or more resources.

89. A method as recited in claim 83, wherein nodes in said bounded path of nodes are all nodes in the graph that are bounded between start nodes and a stop node.

90. A method as recited in claim 84, wherein nodes in said bounded path of nodes are all nodes in the graph that are bounded between a set of start nodes and a stop node.

91. A method as recited in claim 90, wherein all root nodes in said invocation graph are members of the start nodes.

92. An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 1.

93. An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing identification, the computer readable program

code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 29.

94. An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 50.

95. An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing statistical detection, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 80.

96. A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 15.

97. A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 32.

98. A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing identification, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 44.

99. A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing identification, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 47.

100. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for identification, said method steps comprising the steps of claim 1.

Page 43 of 50